Smart Autonomous Vehicles in High Dimensional Warehouses Using Deep Reinforcement Learning Approach

Mohamed Rhazzaf and Tawfik Masrour, *Member, IAENG,*

***Abstract*—In this paper, we propose a smart planning and control system for autonomous vehicles in a high dimensional space. It is a complete unsupervised scheduler and motion planner. Many warehouses take advantage of using an automated material handling process for product transshipment to speed up procedures. However, the growth of the space dimensions becomes a big issue for the control system that becomes increasingly complex. The introduced model uses, as input, a kernel with a control system based on Deep Reinforcement Learning for the low dimensional space. Besides, it employes a global transition-control system to intelligently coordinate communications between the kernels. The global transition-control system creates virtual paths for each product, assigns tasks to kernels for handling products in their zones, and ensures transitions between blocks to brings each product to their destination. Our approach yields good performance in terms of speed and number of movements. The system is robust to the increase, as well, the size of the warehouse as the number of products.**

***Index Terms*—autonomous vehicles, deep reinforcement learning, multi agent system, automated guided vehicle system, material handling, planing design**

I. Introduction

With the beginning of the 21^{st} century, the involvement of artificial intelligence methods in dealing with real-world problems has been massively revealed. Starting with the improvement and the hybridization of several algorithms, like the mixing of the swarm intelligence and machine learning ([1], [2]), or the enhancement of the multi-agent reinforcement learning algorithm ([3],[4]). Moreover, the direct applications of artificial intelligence in the manufacturing ([5],[7]), management ([6]), and security ([8]).

In recent years, the automation process has had a central interest in the industrial sector. As a manifestation of Industry 4.0 [9], which present the trending and the bridge to the use of artificial intelligence in the industrial process, several requirements related to automation. Besides, the flexibility of many processes are becoming necessary due to the trend of minimizing human intervention in repetitive tasks.

Manufactories have used Automated Guided Vehicles (AGVs) earlier as one of the industrial examples of the repeated process. The AGVs can move automatically, following a path drawn or programmed in advance. They can make a series of scheduled stops and perform different tasks such as pick-up and drop-off items. However, recent developments have lead to new generations of autonomous vehicles equipped with automotive industry technologies such as detection, recognition and anti-collision systems.

Autonomous vehicle decision-making core in a warehouse is, in some sort, an industrial generalization of what it called the mobile robot path planning [10]. It is, in fact, the problem of finding the shortest path according to some criteria, relating two positions in a static environment configuration (shape, obstacles, and so on). Many algorithms can deal with it like [11], [12]. The difference is that we face a dynamic environment with many robots with the purpose of getting the product first, and next finding the shortest path to its destination.

In the context of material handling, the autonomous vehicles decision-making core claims advanced algorithms that manage the assignment of each vehicle to some tasks in addition to resolving the conflict access between them. We describe such a framework as an Intelligent Material Handling System. A kind of system that integrates camera vision, environmental localization and mapping, decision support or decision-making and communication.

The simple way to create a material handling system of a warehouse is to design three independent software: The pathfinder that assign the path to a vehicle ignoring the other dynamic components (the other vehicles), the deadlock detector, and the deadlock solver. However, such systems require advanced algorithms that become complex for high dimension environment and a large number of vehicles.

Our approach to solving material handling in a warehouse can be described by dividing the warehouse into a grid of low dimension zones. Each zone is managed by a control system kernel using the deep reinforcement learning algorithm to pick-up products in their territories. The kernel manages, efficiently, the pathfinder, the deadlock detector, and the deadlock solver systems. A master control system will compute the Cartesian path intersection with borders

Manuscript received Jul 04, 2020; revised Nov 26, 2020. This work was supported in part by Moulay Ismail University, Meknes, Morroco.

Mohamed Rhazzaf is PhD student at Laboratory of Mathematical Modeling, Simulation and Smart Systems (L2M3S), Artificial Intelligence for Engineering Sciences Team (IASI), Doctoral Studies Center, 15290 ENSAM, Moulay Ismail University, 50500 Meknes, Morocco, e-mail: m.rhazzaf@edu.umi.ac.ma

Tawfik Masrour is Professor at Laboratory of Mathematical Modeling, Simulation and Smart Systems (L2M3S), Artificial Intelligence for Engineering Sciences Team (IASI), Department of Mathematics and Computer Science, 15290 ENSAM, Moulay Ismail University, 50500 Meknes, Morocco, e-mail: t.masrour@ensam.umi.ac.ma

and let each kernel control system manage only its zone.

The paper is organized as follows. In section 2, we present a summary of autonomous vehicle works. Section 3 details the proposed model for a high dimensions warehouse by beginning with highlighting the basic neural network reinforcement learning approach for a control system in a low dimensions zone, following by its improvement in large dimensions zone. After that, we show the results in section 5. The last section gives some conclusions and perspectives.

II. Related works

In literature, Automated Guided Vehicle (AGV) was the first implementation of the concept of the autonomous vehicle in the industry. It is a driver-less robot moving products or objects in two dimensions space that navigates using physical guide paths, such as buried wire or magnetic tape. The second generation of autonomous vehicle uses a wireless guidance system instead of physical support. With the progression of the computer vision and deep learning algorithms, the autonomous vehicle utilizes new techniques like the vision-based guidance system, allowing more flexibility and implementation simplicity.

Many architectures are proposed to deal with the autonomous vehicle. [13], for example, combines three navigation modules that helps the system following the expressed path. [14] proposes a navigation system implementing data aggregation to integrate all information collected from many sources like camera, odometer, LIDAR, and collision avoidance sensors. Other approaches implement more sophisticate artificial intelligence algorithms like A* ([15]) and D*([16]), Q-learning algorithm in [17], artificial bee colony algorithm and an evolutionary programming algorithm in [18].

Studying the Autonomous vehicles system requires a test environment in order to compare proposed methods. One of the classic approaches is to transform the environment into a grid map and find the shortest path without obstacles. Moreover, task assignment problem, in comparison with navigation and trajectory planning problems, little research has been focused on it. Different architecture can manage it, we cite according to [19] the rule of random vehicle, the nearest vehicle, the farthest vehicle, the most chosen, and the last chosen vehicle. However, for the deadlock management ([20], [21], [22]), there are three main designs. The first one is the pure hardware design, which is described as a specific physical architecture for the environment to avoid overlapping between vehicles. The second one is the soft-hard design. It's a control system that requires the presence of one vehicle in a predefined zone during the process to avoid deadlock and collision. And the last one is the pure software design, which presents an entire algorithm to control deadlock with no consideration of the environment architecture.

Other methods that deal with autonomous vehicles use a deep reinforcement learning approach. Reinforcement learning (RL) [23] is a branch of artificial intelligence which deals with the learning of agents who evolve in an environment by performing a certain number of actions. Learning is done, through trial and error, with the objective of optimizing a reward function. This paradigm is used mostly in several industrial contexts like [24]. But unlike the other techniques, reinforcement learning doesn't need labeled input/output pairs used in the supervised learning, and maximize a reward signal instead of trying to find hidden structure, unlike the unsupervised learning. However, Deep reinforcement learning mixes the RL with the usage of a neural network to overcome the complexity of the problem. For example, [25] develops a deep reinforcement learning technique so that the agent learns to choose and then move towards the closest objective among the multiple possible tasks. Another method, using deep reinforcement learning, to control several AGVs was proposed by [26]. [27] proposed a way to train the vehicle agent to learn an automated lane change behavior such that it can intelligently make a lane change under diverse and even unexpected scenarios.

III. Deep reinforcement learning control system for an Autonomous Vehicle System in large dimensional warehouses

Before we present our model of the high dimensional warehouse control system, we should first detail the basic framework expressed in [28]. It is, in fact, the core kernel used in the new proposed model in order to overcome the above presented issues.

A. Deep reinforcement learning control system of an Autonomous Vehicle System

To construct a pure soft design for an autonomous vehicle control system, several inputs and considerations should first be defined. As examples, for vehicles, we have the number of free robots in the environment, their positions, and their accessibility to the untreated products. For products, we specify their information, their localization, and their accessibility to the entire set of vehicles. After that, we should design an algorithm that can assign vehicles to products and indicates the paths that avoid deadlock situations and collisions in the environment during the process.

According to [28], we can bypass all the complexity of implementing algorithms for the control system by using a neural network reinforcement learning approach. The idea is to model the control system as a Reinforcement Learning problem using Markov Decision Process [23] with the set of vehicles as one big agent, interacting with the environment by attributing actions to all vehicles in the environment, with a reward system that promotes the collection of products in minimum steps and of course avoiding the deadlock situations.

The implementation of [28] illustrated at fig 1 uses the Q-Learning [29] approximation approach and the D3QN (Dueling Double Deep Q-Network) [30] algorithm for the learning process. The training process is based on a reward system described in the table I, giving reward values for each

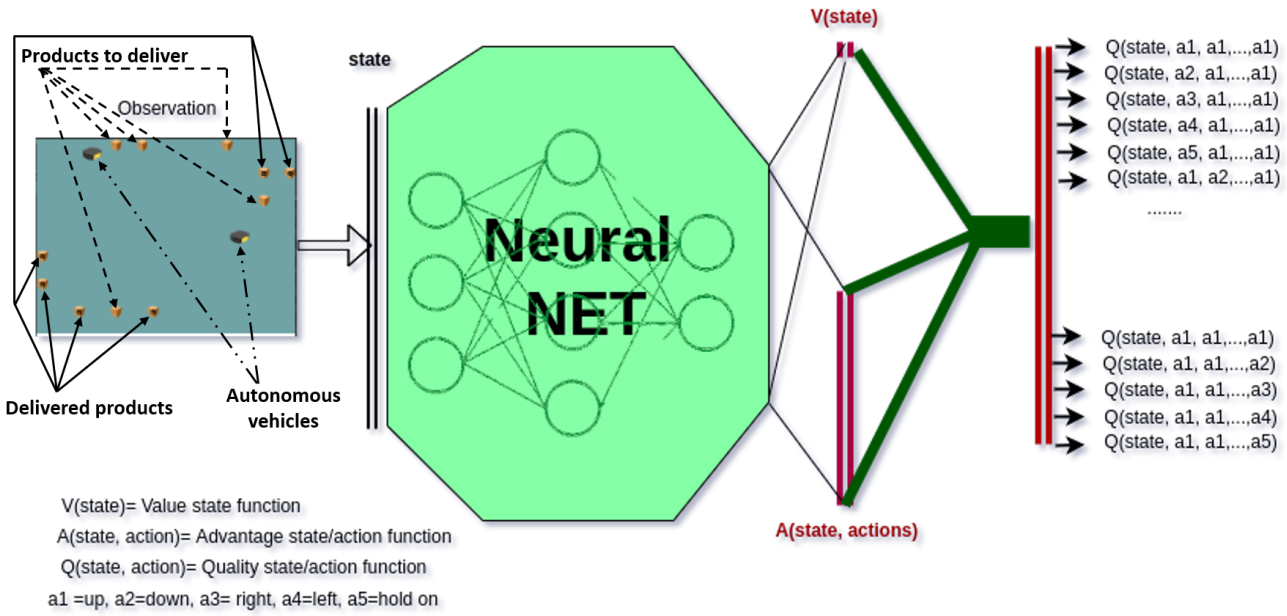


Fig. 1: Deep reinforcement learning control system kernel architecture. We use the state data extracted from the environment (vehicle positions, vehicle status, product positions, product status), and we feed-forward them in a 3DQN architecture to get Q values for each possible action. The best action to take is with the high Q value.

TABLE I: Kernel Reward system

Role	Reward
staying without product	-0.08
staying with product	-0.2
collision with border without product	-0.2
collision with border with product	-0.4
collision with vehicle without product	-0.6
collision with vehicle with product	-0.3
stand by without product	-0.1
stand by with product	-0.2
loading product	0.8
delivered product	1

vehicle dispatched in the environment. The big agent will get in each step the mean of all collected rewards. Besides, The progression learning procedure applies a smooth dimensions augmentation, using the transfer learning techniques between each dimension transition.

The results show a significant performance with a small number of vehicles. But with the growth of its number, the action space increases exponentially, which drops the system performance. It should also be mentioned that, as the environment dimensions increase, the model loses its performance progressively, due to the simplicity of the policy proposed by the neural network. So as a problem to solve, we need an approach to increase the environment dimensions considerably and maintain at the same time a good performance with such a control system model, which we will propose in the next section.

B. Proposed model for high dimensions environment control system

The proposed model contains three parts: path recognizer and kernel input extractor, the transition system, and the kernel control system.

Let us consider L a high dimensions environment of an autonomous vehicle system with dimensions (L_0, L_1). To decompose this environment into blocs of small zones, we need a kernel model with neural network reinforcement learning control system, and a small number of vehicles. That kernel k should be of dimensions $[\frac{L_0}{n_0}, \frac{L_1}{n_1}]$ to get a rectangle grid of elements of it, with size (n_0, n_1).

Let's consider a product in the environment L, with a start position (i_0, j_0) and a destination position (i_1, j_1).

First, to build the path recognizer and kernel input extractor, we define a virtual minimum path linking the two positions of the product. Since we have a convex shape environment, we can consider the minimum distance between two points as the distance of the segment connecting them. So the virtual minimum path, for the reason of the discrete environment space, is the oriented Manhattan path from the start to the destination positions.

We define the borders of a kernel as the set of all the positions in the borders.

We extract all the borders' positions that intersect with the Manhattan path. We will have two situations for each kernel:

- **Kernels that have not intersections with the Manhattan path of the product:** These kernels will not participate in delivering the product.
- **Kernels that have one or two intersections with the Manhattan path of the product:** These kernels will take over the delivery of the product. However, each of them needs the start and the end positions with respect to the relative landmark. So we will mix up the start and destination positions for kernels with one intersection, and define the two intersection positions as the start

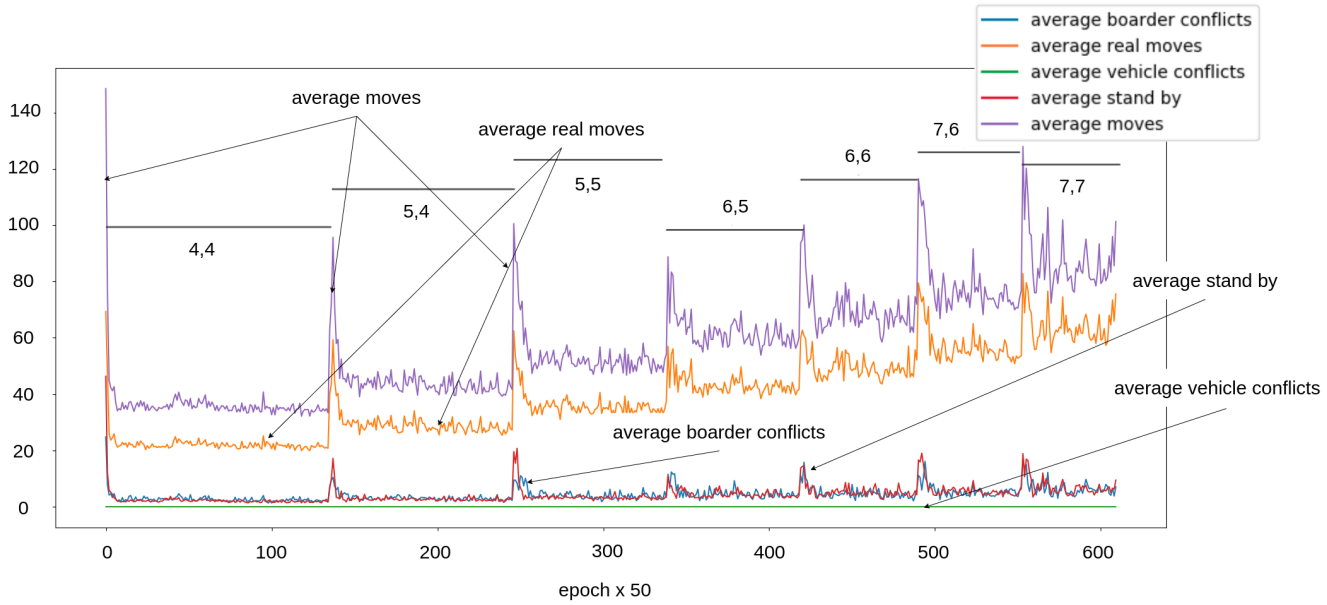


Fig. 2: The Performance of the autonomous vehicle system in the learning stage. The learning process starts with training the agents with a 4x4 grid size until it gets a stable result. After that, it uses the saved model to tackle the 4x5 grid size to get another stable result, and so on until we reach the 7x7 grid size. The spikes figured out between two environment size configuration indicate the model disturbances at the beginning related to the environment configuration modification.

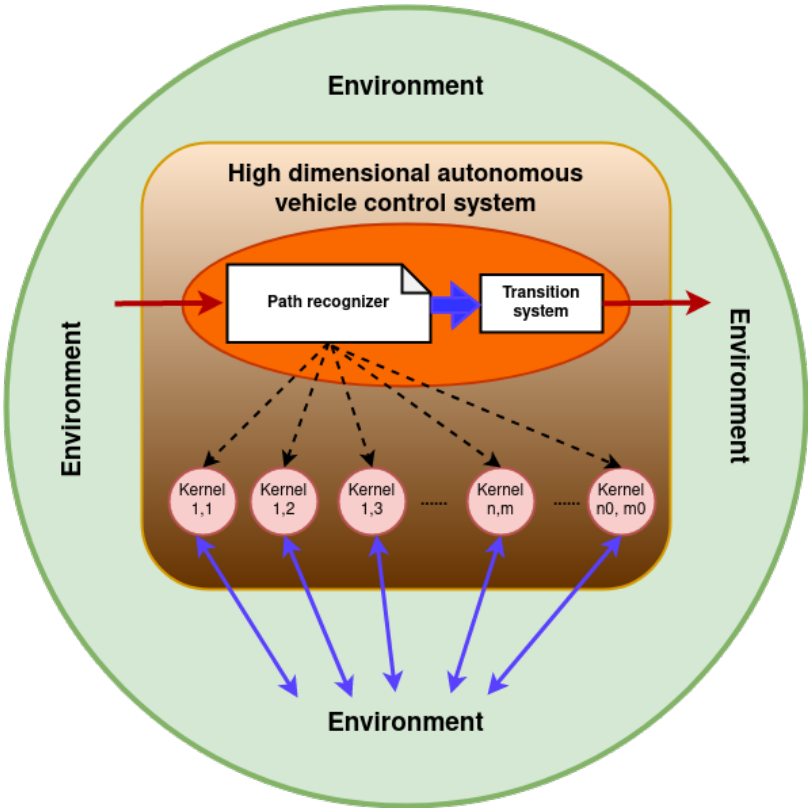


Fig. 3: High dimensional autonomous vehicle control system architecture

and the destination positions for the second type of kernels, with respect to the orientation Manhattan path. So as result, we will have p oriented kernels $\{(K_1, S_{K_1}, D_{K_1}), (K_2, S_{K_2}, D_{K_2}), ..., (K_p, S_{K_p}, D_{K_p})\}$ where K_i is the i^{th} kernel position, and S_{K_i}, D_{K_i} are respectively the start and the destination positions where the vehicles in i^{th} kernel will act.

Second, to construct the transition system between kernels, which present technically a system picking up product from the destination position of the kernel $k-1$ and the start position of the kernel k, we need a pickup system to deliver a product from the destination kernel position to the desired adjacent start kernel position.

Finally, we need a kernel control system that takes advan-

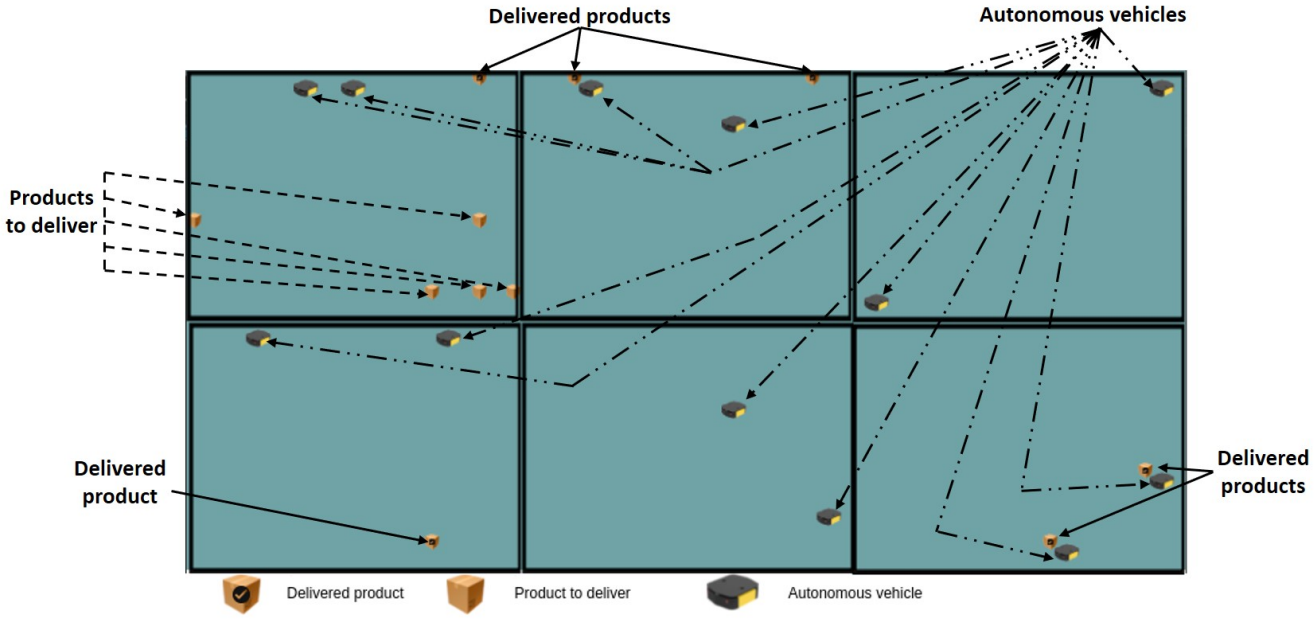


Fig. 4: Decomposition of Autonomous Vehicle environment into grid of kernels

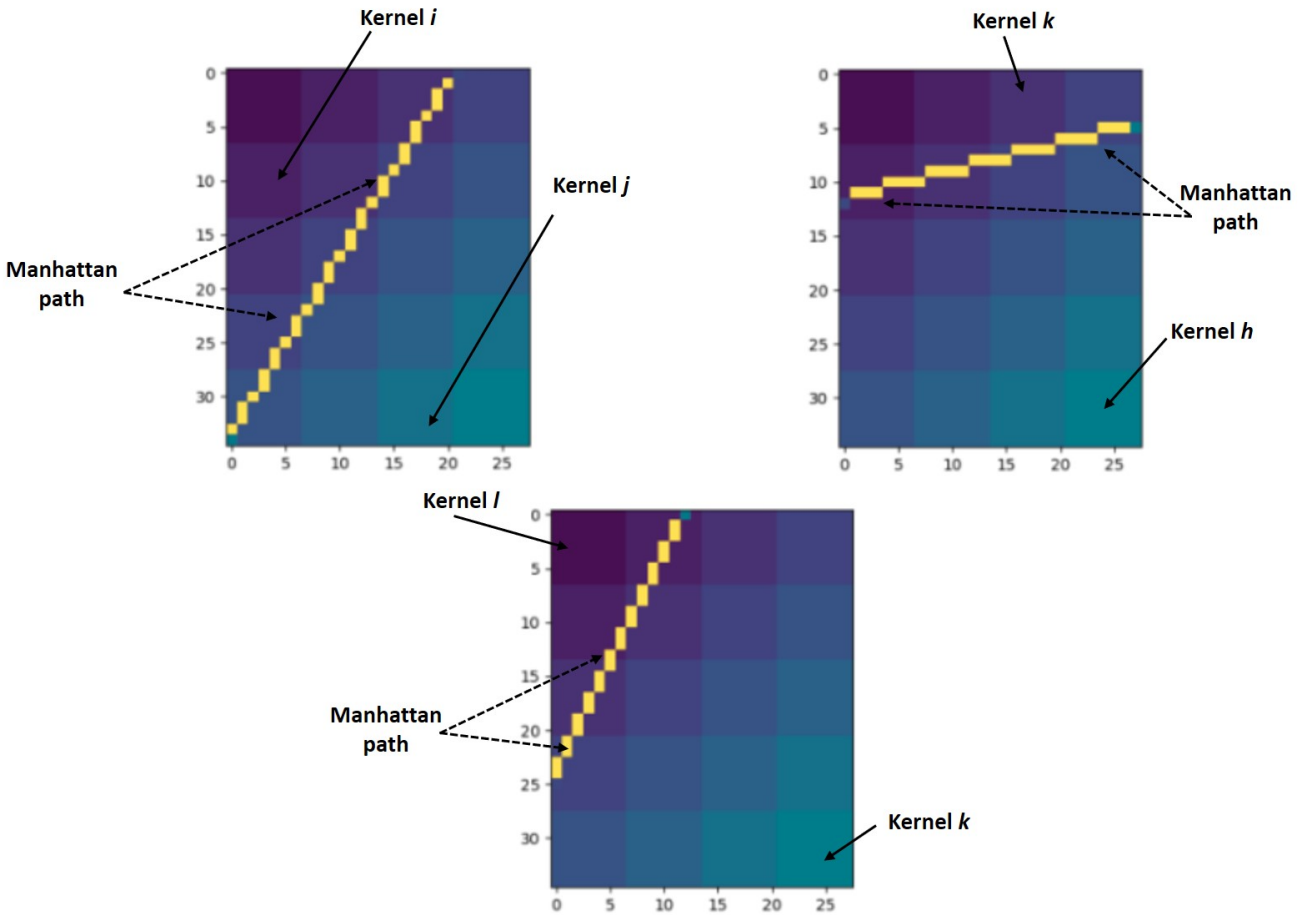
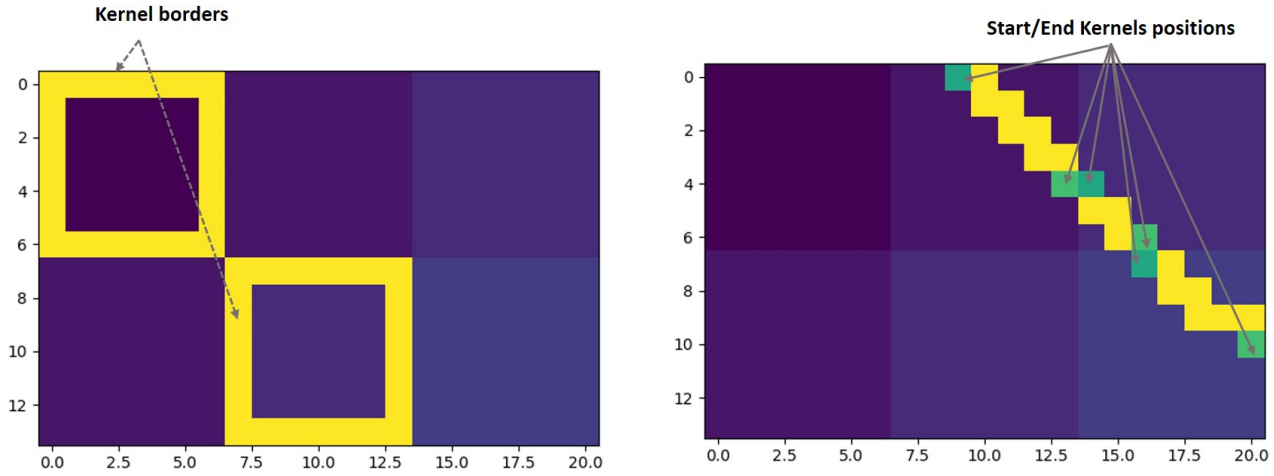


Fig. 5: Manhattan paths for 3 random start/destination product positions. The presented environments are subdivided into kernels (degraded dark regions) and the Manhattan path (light regions) which is determined with a start location and a destination location..

tage of managing products inside its territory, which is in our case the neural network reinforcement learning control system.



(a) (b)

Fig. 6: (a) Example of kernel borders for an warehouse environment, (b) Start/end kernels positions for a product Manhattan path

Algorithm 1: Autonomous-Vehicle-Control-System

```
Result: no result
Initiate-Agents-Kernel-Positions()
while True do
    if Has-Product-Delivering() then
        Start, End = Start-End-Product-Positions()
        Position-List = Manhattan-Path(Start, End)
        Borders-Kernels-List = Get-All-Kernels-Border()
        Start-End-Kernel-List = Get-Start-End-Borders-Kernels(Position-List, Borders-Kernels-List)
        while Start-End-Kernel-List is not Empty do
            Kernel-Position, Start-Position, End-Position = Get-Information(Start-End-Kernel-List[0])
            Call-Kernel-Control-System(Kernel-Position, Start-Position, End-Position)
            if Product-Position == End-Position in Kernel-Position landmark then
                Old-End-Position =End-Position
                Delete(Start-End-Kernel-List[0])
                Kernel-Position, Start-Position, End-Position = Get-Information(Start-End-Kernel-List[0])
                Product-Transition(Old-End-Position, Start-Position)
            else
                Wait()
            end
        end
    else
        Wait()
    end
end
```

IV. Simulation and Results

Our model has been simulated with a pre-trained 7x7 size environment kernel with 2 agents and 5 products to deliver, using a pseudo greedy policy for the control system with 15% of exploration. The performance of our approach will be judged on the set of kernel grid sizes (1,2), (2,2), (2,3), (3,3). The construction of the kernel is made with the procedures described in [28], and the source code of the simulation is accessible in [31]

Our approach will be evaluated by making a comparison with random policy. It presents the same environment architecture with kernels that adopt a control system with a fully random decision policy. The two policies will be tested in two environments that start with 10 products with random start and destination positions located in the environment borders.

A video depicting a chronological set of the warehouse state in the simulation of picking up five products with a 3x2 grid environment using our approach can be viewed here [32].

For more clarification, the figure 7 details the full path of each vehicle inside a 2x2 grid warehouse environment with five products. Each product path will mentioned with different color. The thin arrows present the transition inside

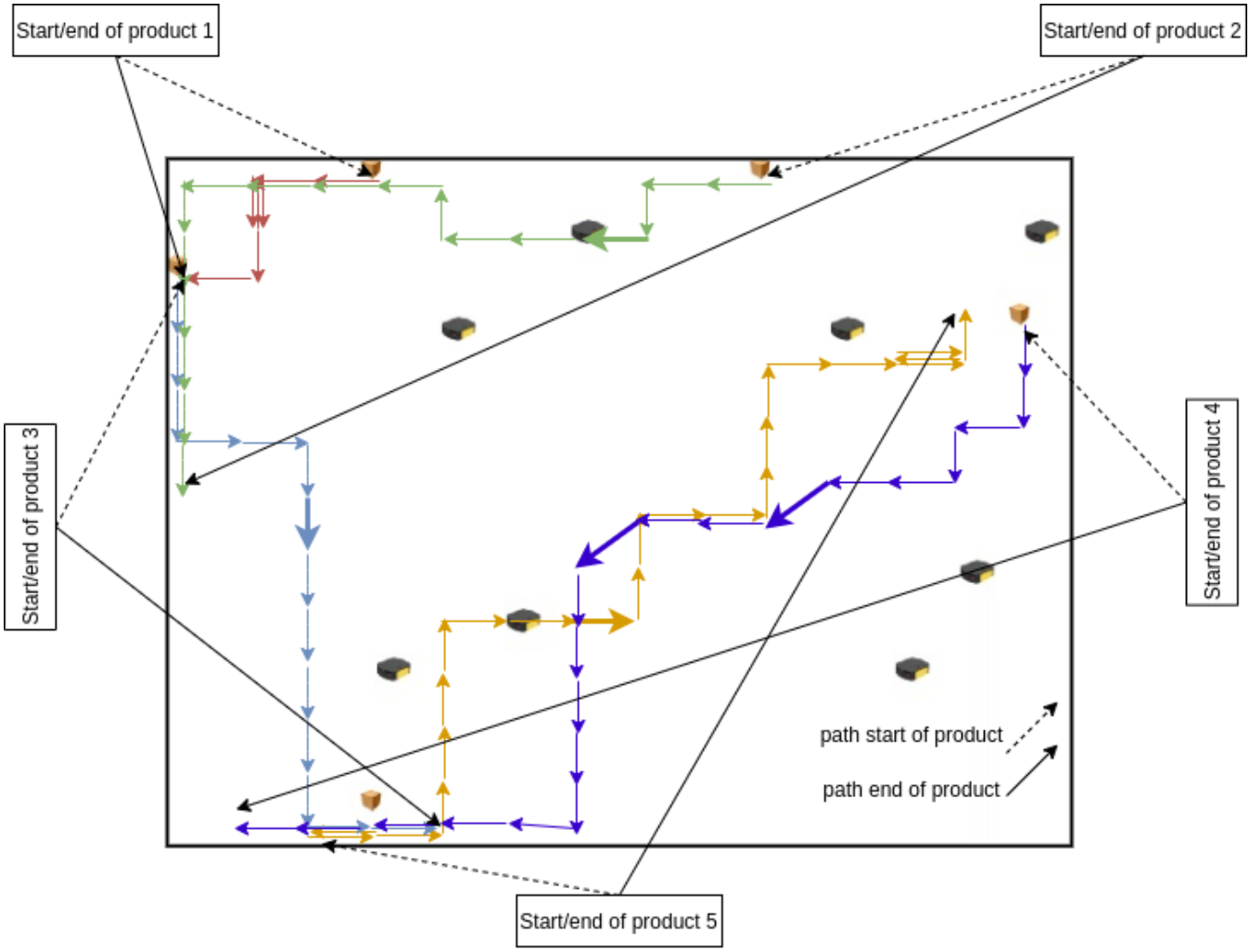


Fig. 7: Simulation of picking up products by autonomous vehicle system in two by two kernel warehouse environment. In this figure, we have the purpose of delivering five products. Five oriented paths are presenting the product picking up in the warehouse with different colors, each one concerns one product. The fat arrows indicate the transition between kernels.

kernels. Moreover, the fat arrows indicate transition between kernels.

For the purpose of evaluating our model, we will use quantitative criteria that compare the model result with the ideal scenario that can be get. In our situation, the best path is the Manhattan one without conflict access or deadlock. But we will consider the worst-case giving that the initial vehicle position is exactly the product destination. In other words, the vehicle should take the Manhattan path to pick up the product, and another time using a different Manhattan path to reach the destination. On the other hand, in order to evaluate our model, we simply get the number of moves made by all the vehicles in the warehouse.

Let's consider N_p, N_v respectively the number of products, the number of vehicles, $H(i)$ the Manhattan path of i^{th} product giving it start and end position, and $S(j)$ is the number of moves of the j^{th} vehicle. We will use the ratio:

$$R = \frac{\sum_{j=1}^{N_v} S(j) - \sum_{i=1}^{N_p} 2H(i)}{\sum_{i=1}^{N_p} 2H(i)}$$

as a metric to compare the two approaches.

The results mentioned in the figure 9 show a great performance for our approach. Firstly, compared with the random approach, we can see a mean ratio between 1 and 2 for our model, versus mean ratio between 20 and 60 for random approach, which can be estimated as $\frac{|2-60|}{60} = \mathbf{96}\%$ our model is better than the random policy.

The large gap between the random policy and the model policy maintains the same intensity in the four graphs, even they present different grid size (2x1, 2x2, 3x2, 3x3). The figure shows that the random policy uses 30 times more of moves compared to our model.

So as result, our model guaranties a better and a stable performance comparing to the random policy, giving an unknown grid configuration.

By analyzing the figure 8, we can observe a smooth rise of the mean ratio values referring to the grid size growth, and that due to the pseudo greedy policy managing the kernel control system. The 15% exploration characterized the kernel control system policy can increase the error rate as we enlarge the grid size.

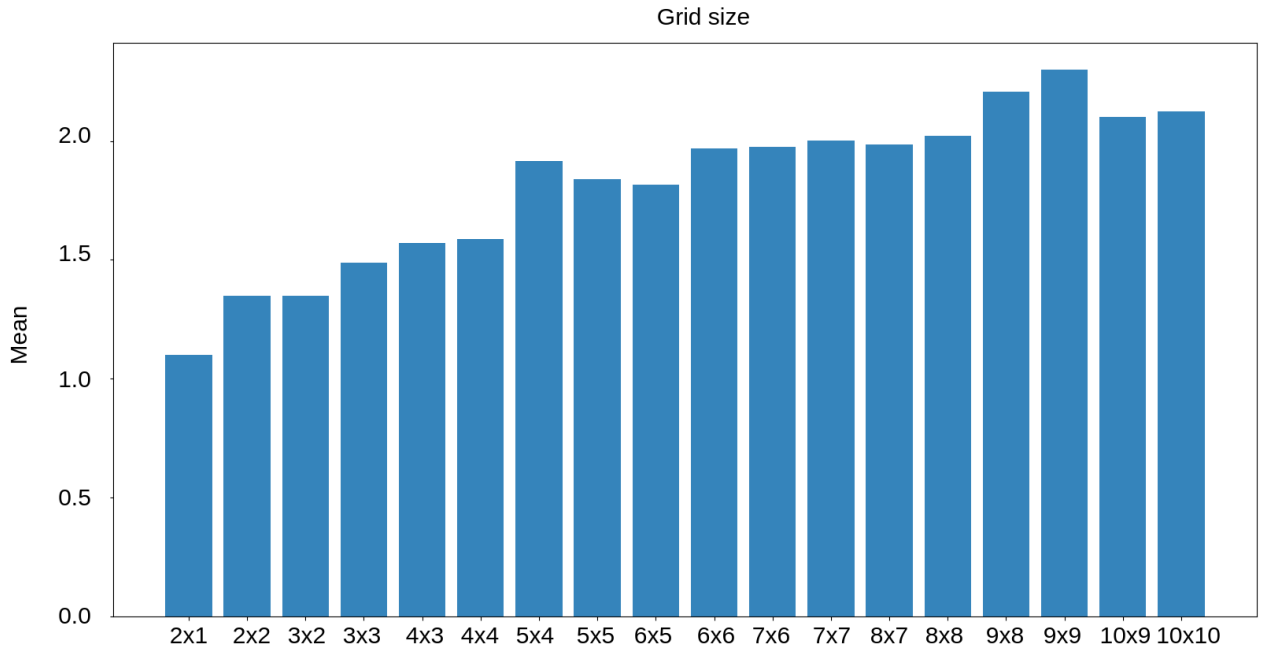


Fig. 8: Mean of R ratio values of autonomous vehicle design of 100 epochs with set of grid sizes. We can figure out a smooth increase of the R values within the dimension size growth due to, in fact, the impact of the exploration rate applied in the simulation.

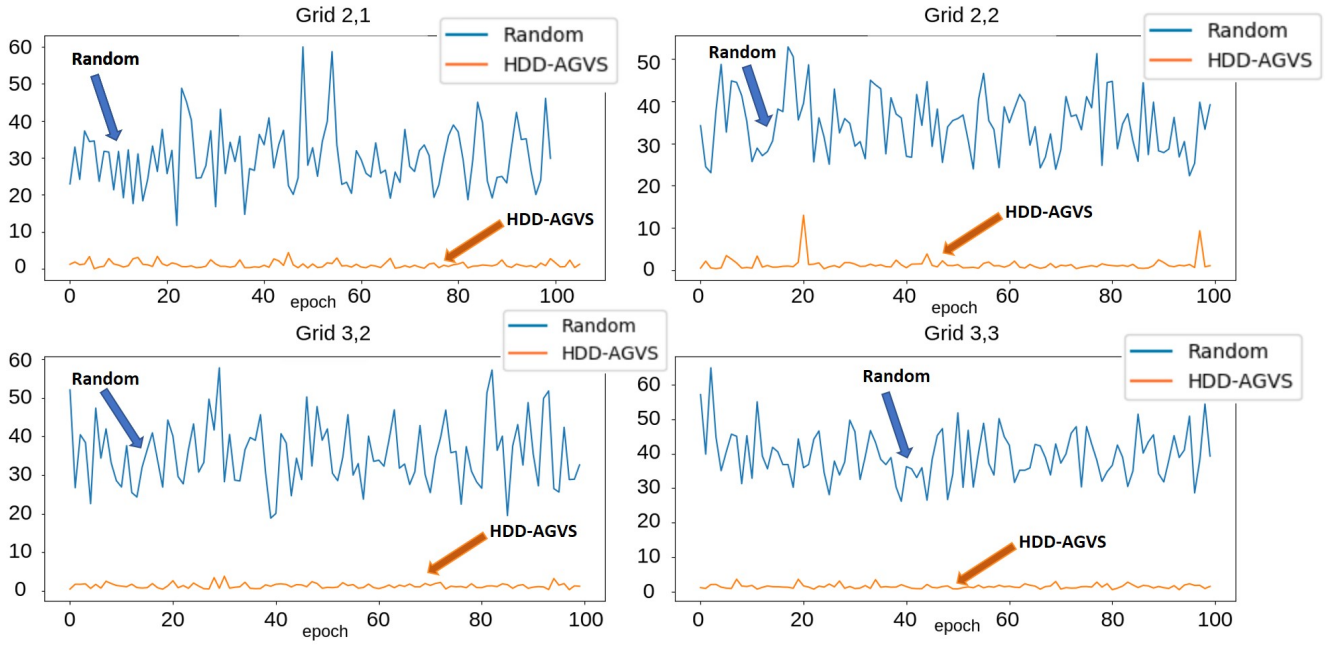


Fig. 9: Ratio values of autonomous vehicle design versus random design in 1x2, 2x2, 3x2, and 3x3 grids in 100 epochs. The four figures mention the R ratio values for the random policy and the proposed model policy.

V. Conclusion

In summary, we can say that our design approach provides a stable alternative ensuring a good yield for high dimensional warehouse with a convex environment shape.

Our approach uses simple implementation mixing the low dimension kernel with a pre-trained control system and a transition rule between kernels

.

However, such an approach requires the presence of agents in each kernel in the warehouse environment, which can be expensive as material in the case of using a high number of kernels.

As perspectives, we can tackle two main issues:

- The minimization of the number of vehicles:

  In fact, we can add another control system dedicated to making the vehicles transition between kernels, and thus authorize it movement with no restriction in the hole warehouse environment.

- The non-convex warehouse environment shape:

  In this case, we cannot use the Manhattan path to describe the product transition, so we can improve the transition system in order to find first the feasible total path, next it will solve the transitions to delegate the tasks to the desired kernels.

REFERENCES

[1] Masrour, T., Rhazzaf, M. (2018). A New Approach for Dynamic Parametrization of Ant System Algorithms. International Journal of Intelligent Systems and Applications (IJISA), 10(6), 1-12.

[2] Rhazzaf, M., Masrour, T. (2020) "A dynamic configuration with a shared knowledge center for multi objective ant colony optimization algorithms". International Journal of Intelligent Systems Technologies and Applications, 2020 Vol.19 No.6, pp.541 - 554.

[3] Vinyals, Oriol, et al. (2019) "Grandmaster level in StarCraft II using multi-agent reinforcement learning." Nature 575.7782 (2019): 350-354.

[4] Amhraoui, E., Masrour, T. (2020). New approach for multi-agent reinforcement learning. In Artificial Intelligence and Industrial Applications - Smart Operation Management. Advances Intelligent Systems and Computing. vol 1193, pp 263-275. Springer, Cham.

[5] El Mazgualdi, C., Masrour, T., El Hassani, I., Khdoudi, A. (2020). A Deep Reinforcement Learning (DRL) Decision Model For Heating Process Parameters Identification In Automotive Glass Manufacturing. In Artificial Intelligence and Industrial Applications - Smart Operation Management. Advances Intelligent Systems and Computing. vol 1193, pp 77-87. Springer, Cham.

[6] Khdoudi, A., Masrour, T., El Mazgualdi, C. (2019). Using Machine Learning Algorithms for the Prediction of Industrial Process Parameters Based on Product Design., In Advanced Intelligent Systems for Sustainable Development. Advances in Intelligent Systems and Computing., vol 1104, pp 728-749. Springer, Cham.

[7] El Mazgualdi, Masrour, T., El Hassani, I., Khdoudi, A. (2020). Machine Learning for KPIs Prediction: A case study of the Overall Equipment Effectiveness within the automotive industry, Soft Computing, Springer.

[8] Hajji, T., Masrour, T. et al. (2020). Distributed and Embedded System to Control Traffic Collision Based on Artificial Intelligence. (2020). In Artificial Intelligence and Industrial Applications - Smart Operation Management. Advances Intelligent Systems and Computing. vol 1193, pp 173-183. Springer, Cham.

[9] B. Paiva Santos, F. Charrua-Santos, and T.M. Lima, "Industry 4.0: An Overview," Lecture Notes in Engineering and Computer Science: Proceedings of The World Congress on Engineering 2018, 4-6 July, 2018, London, U.K., pp415-420

[10] Sariff, N., & Buniyamin, N. (2006, June). An overview of autonomous mobile robot path planning algorithms. In 2006 4th student conference on research and development (pp. 183-188). IEEE.

[11] Cheng, Y. H., Chao, P. J., & Kuo, C. N. "Mobile Robot Path Planning using a Teaching-Learning-Interactive Learning-Based Optimization". IAENG International Journal of Computer Science, vol. 46, no. 2, pp199-207, 2019

[12] A Fast Path Planning Algorithm for a Mobile Robot Adamu, Patience I. and Okagbue, H. I. and Oguntunde, P.E. and Opanuga, A. A. (2018) A Fast Path Planning Algorithm for a Mobile Robot. In: Lecture Notes in Engineering and Computer Science Proceedings of The World Congress on Engineering, July 4-6, 2018, London, U.K.

[13] Fang Q, Xie C (2004). A study on intelligent path following and control for vision-based automated guided vehicle. In Fifth World Congress on Intelligent Control and Automation (IEEE Cat. No. 04EX788) (Vol. 6, pp. 4811-4815). IEEE.

[14] Ramer C, Sessner J, Scholz M et al (2015). Fusing low-cost sensor data for localization and mapping of automated guided vehicle fleets in indoor applications. In 2015 IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI) (pp. 65-70). IEEE.

[15] Hart P E, Nilsson N J, Raphael B (1968). A formal basis for the heuristic determination of minimum cost paths. IEEE transactions on Systems Science and Cybernetics, 4(2), 100-107.

[16] Stentz A (1997). Optimal and efficient path planning for partially known environments. In Intelligent unmanned ground vehicles (pp. 203-220). Springer, Boston, MA.

[17] Jeon S M, Kim K H, Kopfer H (2011). Routing automated guided vehicles in container terminals through the Q-learning technique. Logistics Research, 3(1), 19-27.

[18] Contreras-Cruz M A, Ayala-Ramirez V, Hernandez-Belmonte U H (2015). Mobile robot path planning using artificial bee colony and evolutionary programming. Applied Soft Computing, 30, 319-328.

[19] Egbelu P J, Tanchoco J M (1984). Characterization of automatic guided vehicle dispatching rules. The International Journal of Production Research, 22(3), 359-374.

[20] Viswanadham N, Narahari Y, Johnson T L (1990). Deadlock prevention and deadlock avoidance in flexible manufacturing systems using Petri net models. IEEE transactions on robotics and automation, 6(6), 713-723.

[21] Yeh M S, Yeh W C (1998). Deadlock prediction and avoidance for zone-control AGVS. International Journal of Production Research, 36(10), 2879-2889.

[22] Kim K H, Jeon S M, Ryu K R (2007). Deadlock prevention for automated guided vehicles in automated container terminals. In Container Terminals and Cargo Systems (pp. 243-263). Springer, Berlin, Heidelberg.

[23] Sutton R S, Barto A G (1998). Introduction to reinforcement learning (Vol. 2, No. 4).

[24] Shigei, N., Yamaguchi, Y., & Miyajima, H. "Air Conditioner Control Learning Users' Sensations Based on Reinforcement Learning and Its Scalability Improvement". IAENG International Journal of Computer Science, vol.42, no.3, pp288-295, 2015

[25] Li M P, Sankaran P, Kuhl M E, Ganguly A et al (2018). Simulation analysis of a deep reinforcement learning approach for task selection by autonomous material handling vehicles. In 2018 Winter Simulation Conference (WSC) (pp. 1073-1083). IEEE.

[26] Takahashi K, Tomah S (2020). Online optimization of AGV transport systems using deep reinforcement learning. Bulletin of Networking, Computing, Systems, and Software, 9(1), 53-57.

[27] Wang P, Chan C Y, de La Fortelle A. (2018). A reinforcement learning based approach for automated lane change maneuvers. In 2018 IEEE Intelligent Vehicles Symposium (IV) (pp. 1379-1384). IEEE.

[28] Rhazzaf M, Masrour, T (2020) Deep Learning approach for Automated Guided Vehicle System. In T. Masrour, A. Cherrafi,, I. El Hassani (Eds.), Artificial Intelligence and Industrial Applications - Smart Operation Management. Advances in Intelligent Systems and Computing. Vol. 1193. pp. 227-237, Springer, Cham. Doi:10.1007/978-3-030-51186-9

[29] Watkins C J, Dayan P (1992). Q-learning. Machine learning, 8(3-4), 279-292.

[30] Wang Z, Schaul T, Hessel M, Van Hasselt H et al (2015). Dueling network architectures for deep reinforcement learning. arXiv preprint arXiv:1511.06581.

[31] Rhazzaf M, Masrour T (2020). Source code for a hybrid high dimensional automated guided vehicle system design. Mendeley Data, v1 http://dx.doi.org/10.17632/59g536yght.1

[32] Rhazzaf, M. A. "Smart Autonomous Vehicles in High Dimensional Warehouses Using Deep Reinforcement Learning approach" YouTube, October 31, 2020, . [Video file]. Available: https://www.youtube.com/watch?v=-gFF6458IrY. [Accessed: December 11, 2020].